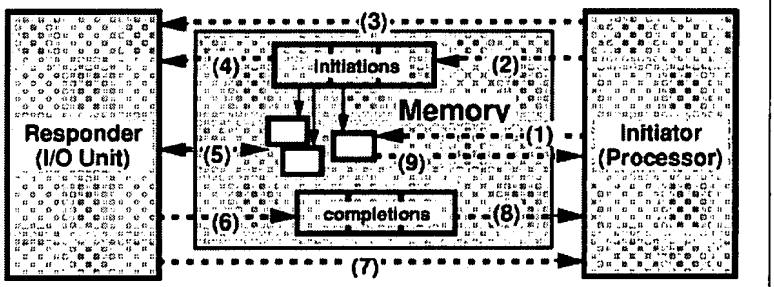
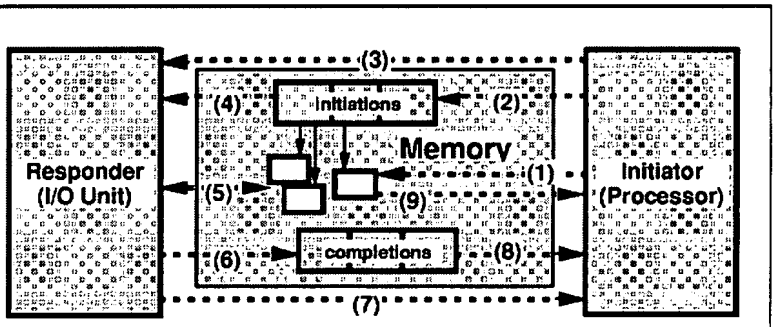
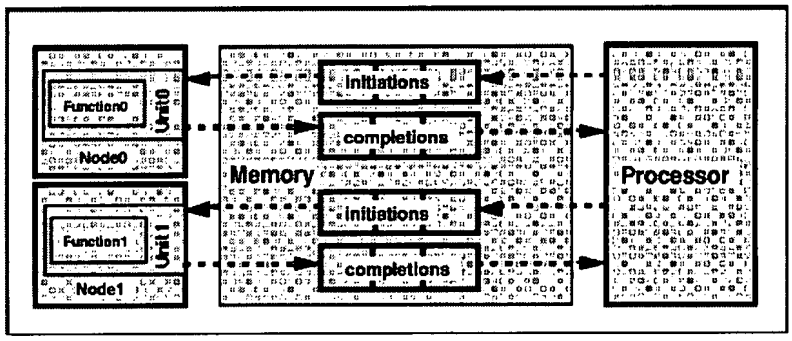
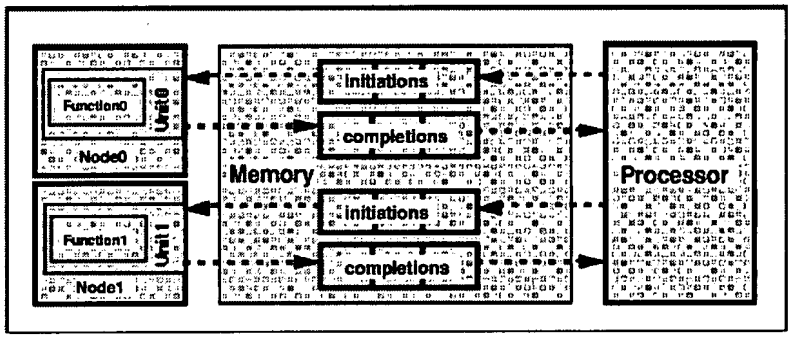
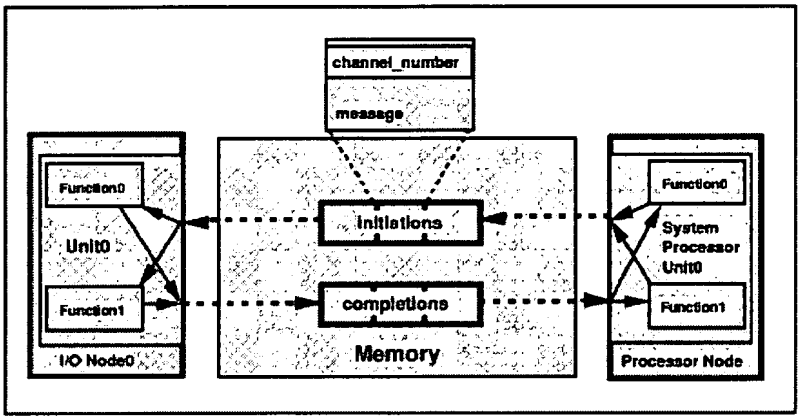
20070501 Exhibit H1 to H6 - Claim Charts for 799.pdf

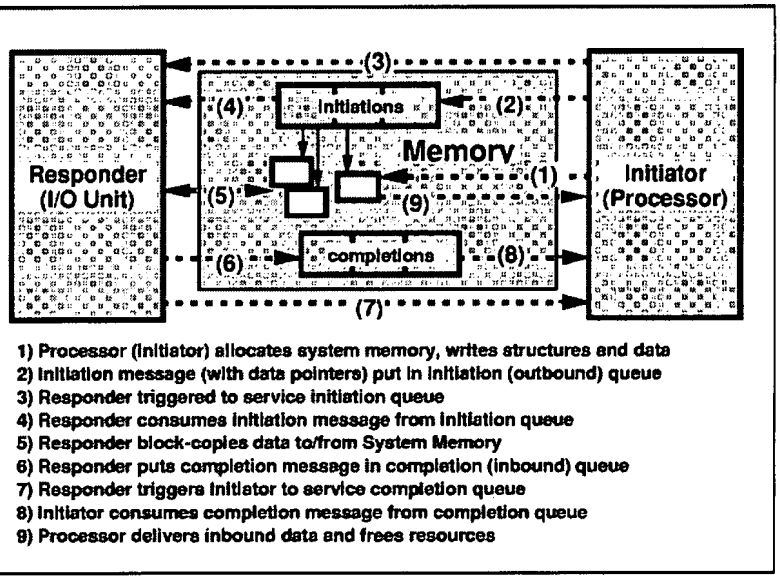
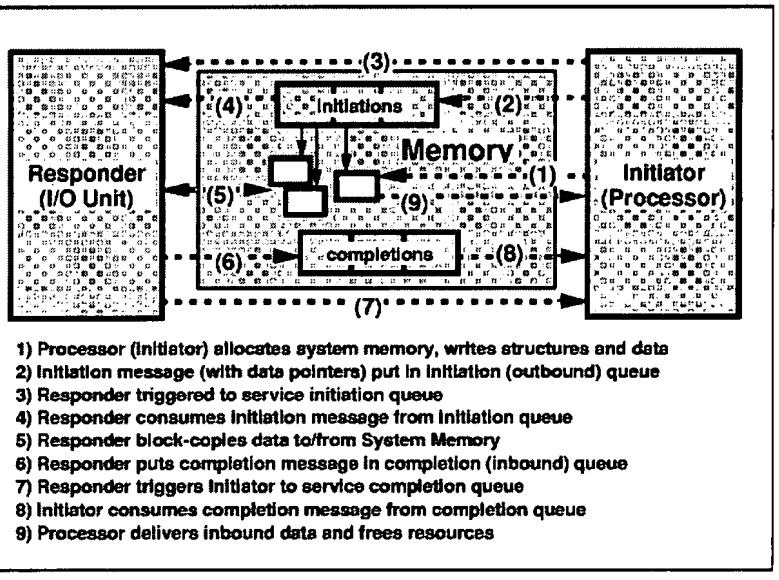
EXHIBIT H-1

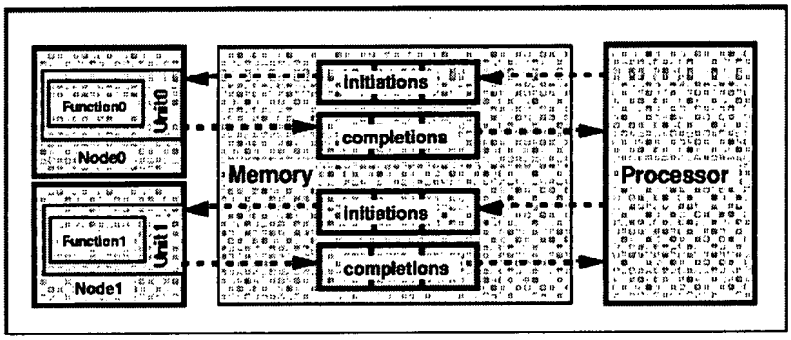
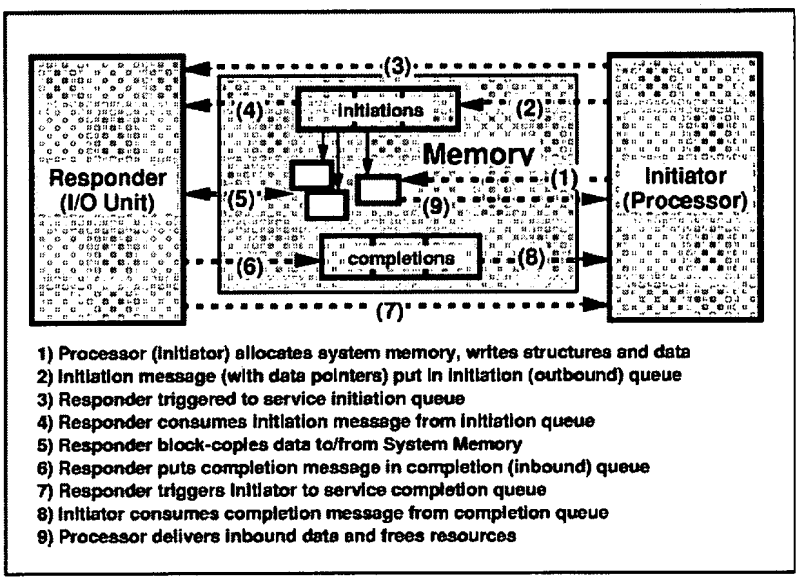
Invalidity Claim Chart for U.S. Patent No. 5,812,799 - IEEE Standard 1212.1

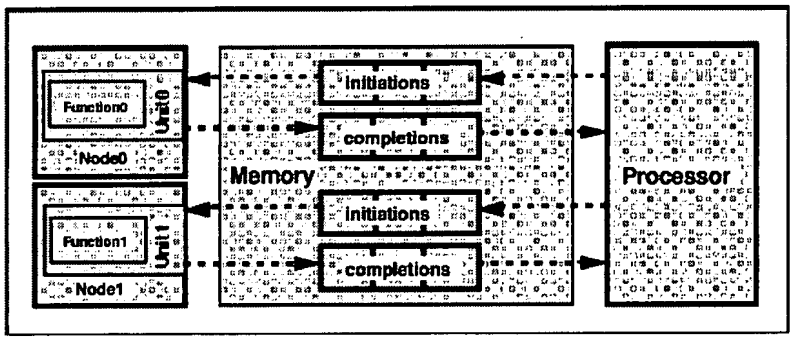
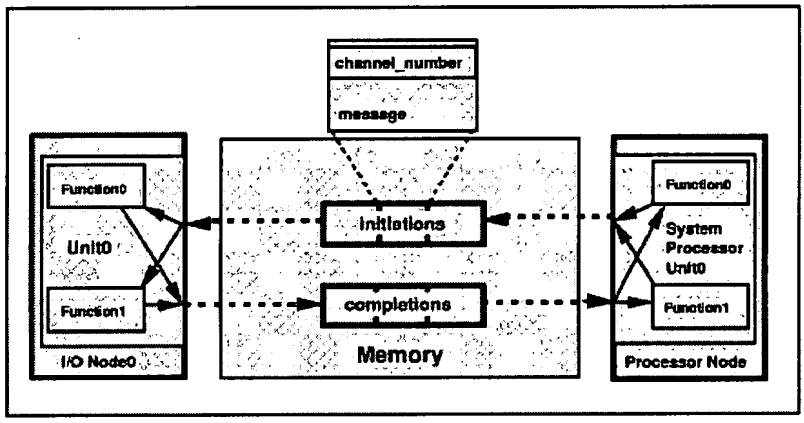
Claim Language	IEEE Standard 1212.1
1. A data processing system comprising:	 1) Processor (initiator) allocates system memory, writes structures and data 2) Initiation message (with data pointers) put in initiation (outbound) queue 3) Responder triggered to service initiation queue 4) Responder consumes initiation message from initiation queue 5) Responder block-copies data to/from System Memory 6) Responder puts completion message in completion (inbound) queue 7) Responder triggers initiator to service completion queue 8) Initiator consumes completion message from completion queue 9) Processor delivers inbound data and frees resources **Figure 1.2—DMA execution model** The *IEEE Standard* describes a data processing system. *See generally, IEEE Standard 1212.1* at 1-5, Fig. 1.2.
one or more requesting processors that generate processor requests directed to one or more peripheral devices;	 1) Processor (initiator) allocates system memory, writes structures and data 2) Initiation message (with data pointers) put in initiation (outbound) queue 3) Responder triggered to service initiation queue 4) Responder consumes initiation message from initiation queue 5) Responder block-copies data to/from System Memory 6) Responder puts completion message in completion (inbound) queue 7) Responder triggers initiator to service completion queue 8) Initiator consumes completion message from completion queue 9) Processor delivers inbound data and frees resources **Figure 1.2—DMA execution model** The *IEEE Standard* discloses a requesting processor (Initiator) that passes messages (i.e., processor requests) to a peripheral device (Responder). *See generally, IEEE Standard 1212.1* at 1-

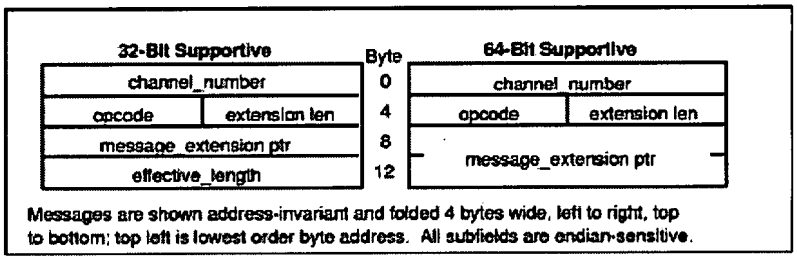
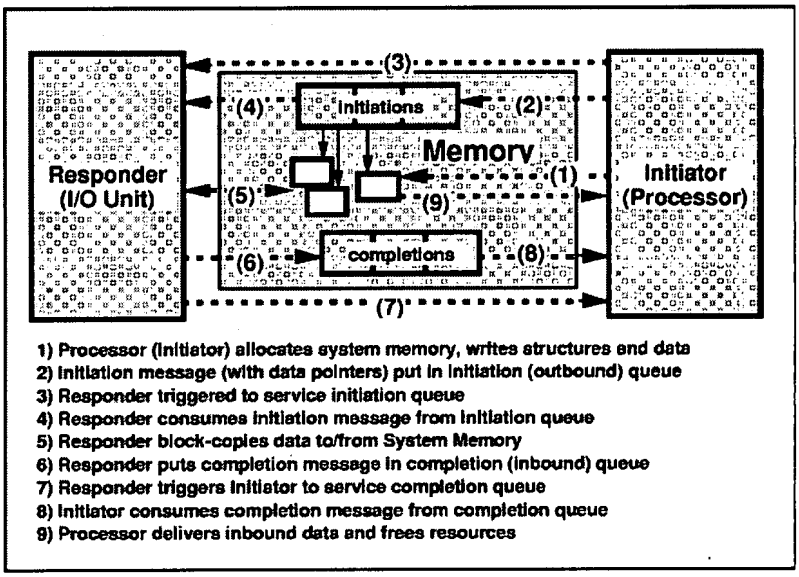
Claim Language	IEEE Standard 1212.1
	5, 46-47, 52, 57-59, Fig. 1.2.
a plurality of peripheral devices that accept the processor requests; and	 **Figure 1.3—One function per node** The peripheral devices (Unit0, Unit1) accept the messages passed from the processor. *See generally, IEEE Standard 1212.1* at 3-7, Fig. 1.3.
a controller responsive to the processor requests that creates a plurality of separate pending queues corresponding to each one of the plurality of peripheral devices for queuing the processor requests directed to a particular peripheral device in entries of a corresponding separate pending queue,	 **Figure 1.3—One function per node**  **Figure 1.5—Multiple functions per unit** The pending queues (Initiations) queue requests from the processor and correspond to each one of the peripheral devices

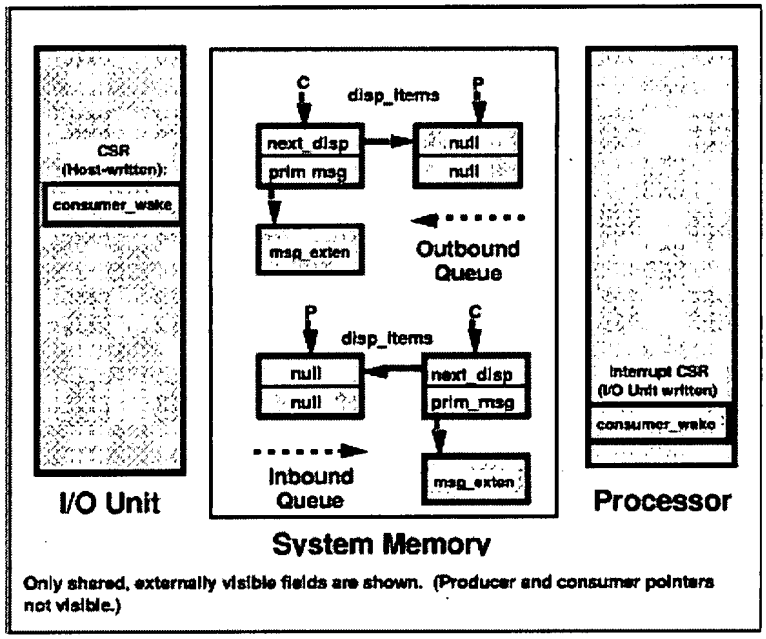
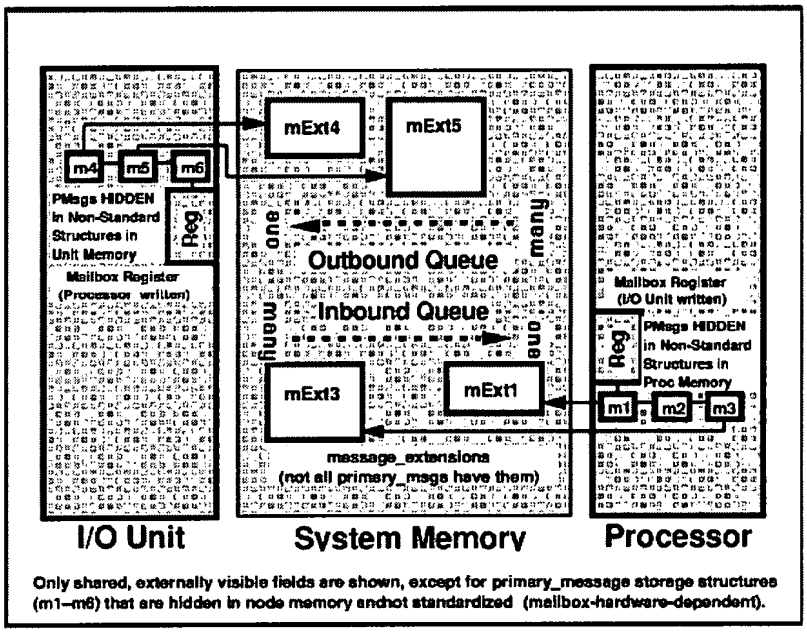
Claim Language	IEEE Standard 1212.1
	(Unit0, Unit1). *See generally, IEEE Standard 1212.1* at 4-7, Figs. 1.3, 1.5. The *IEEE Standard* discloses a DMA controller that manages and provides the DMA models (e.g., circular queue, dispatch list, shareable list, and mailbox models). *See generally, IEEE Standard 1212.1* at 90-92.
wherein at least two separate peripheral devices process the processor requests simultaneously, after retrieving such processor requests from their respective separate pending queues,	The *IEEE Standard* discloses simultaneous processing of the processor requests by the Responders after the Responders consume the initiation messages from the initiations queues. *See generally, IEEE Standard 1212.1* at 2-5, Fig. 1.2.
wherein the one or more requesting processors include dependency checking logic that generate non-blocking processor requests.	The *IEEE Standard* discloses a general purpose processor, which can inherently run software for dependency checking. *See generally, IEEE Standard 1212.1* at 1-3.
3. The data processing system according to claim 1, wherein the non-blocking processor requests are generated by running real-time processes.	The non-blocking processor requests can be for real-time processes. *See generally, IEEE Standard 1212.1* at 10.

Claim Language	IEEE Standard 1212.1
10. A data processing system comprising:	 Figure 1.2—DMA execution model The *IEEE Standard* describes a data processing system (as shown in Figure 1.2). *See generally, IEEE Standard 1212.1* at 1-5.
one or more requesting processors that generate non-blocking processor requests directed to one or more peripheral devices;	 Figure 1.2—DMA execution model The *IEEE Standard* discloses a requesting processor (Initiator) that passes nonblocking messages (i.e., processor requests) to a peripheral device (Responder). *See generally, IEEE Standard*

Claim Language	IEEE Standard 1212.1
	1212.1 at 1-5, 46-47, 52, 57-59, Fig. 1.2.
a plurality of peripheral devices that accept the non-blocking processor requests;	 **Figure 1.3—One function per node** The peripheral devices (Unit0, Unit1) accept the nonblocking messages passed from the processor. *See generally, IEEE Standard 1212.1* at 2-7, Fig. 1.3.
a shared memory device; and	System memory is a shared memory device. *See generally, IEEE Standard 1212.1* at 5, 19, Fig. 1.2.  **Figure 1.2—DMA execution model**

Claim Language	IEEE Standard 1212.1
a memory controller responsive to the non-blocking processor requests that creates a plurality of separate pending queues on the shared memory device corresponding to each one of the plurality of peripheral devices for queuing the non-blocking processor requests directed to a particular peripheral device in entries of a corresponding separate pending queue,	 **Figure 1.3—One function per node**  **Figure 1.5—Multiple functions per unit** The pending queues (Initiations) queue the nonblocking requests from the processor and correspond to each one of the peripheral devices (Unit0, Unit1). *See generally, IEEE Standard 1212.1* at 2, 4-7, Figs. 1.3, 1.5. The *IEEE Standard* discloses a DMA controller that manages and provides the DMA models (e.g., circular queue, dispatch list, shareable list, and mailbox models). *See generally, IEEE Standard 1212.1* at 90-92.
wherein at least two separate peripheral devices process the non-blocking processor requests simultaneously, after retrieving such non-blocking processor requests from their respective separate pending	The *IEEE Standard* discloses simultaneous processing of the nonblocking processor requests by the Responders after the Responders consume the initiation messages from the initiations queues. *See generally, IEEE Standard 1212.1* at 2-5, Fig. 1.2.

Claim Language	IEEE Standard 1212.1
queues,	
wherein the entries include pointers that point to memory locations on the shared memory device.	An entry (transaction-initiation message) includes a pointer to another memory location (message_extension ptr). *See generally, IEEE Standard 1212.1* at 82, Fig. 10.7.  **Figure 10.7—16-byte transaction-initiation messages** Requests can include pointers to memory locations on the shared memory device (System Memory). *See generally, IEEE Standard 1212.1* at 4-5, 50, 62-63, 81, Figs. 1.2, 6.1, 8.1.  **Figure 1.2—DMA execution model**

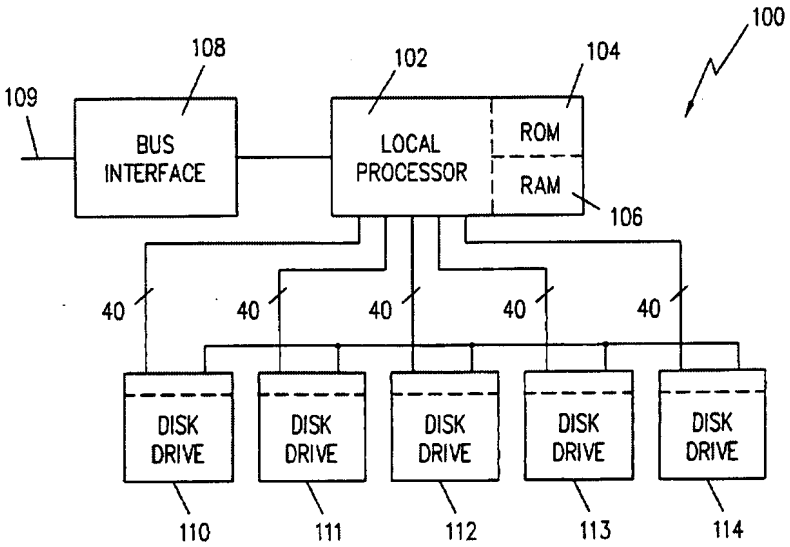
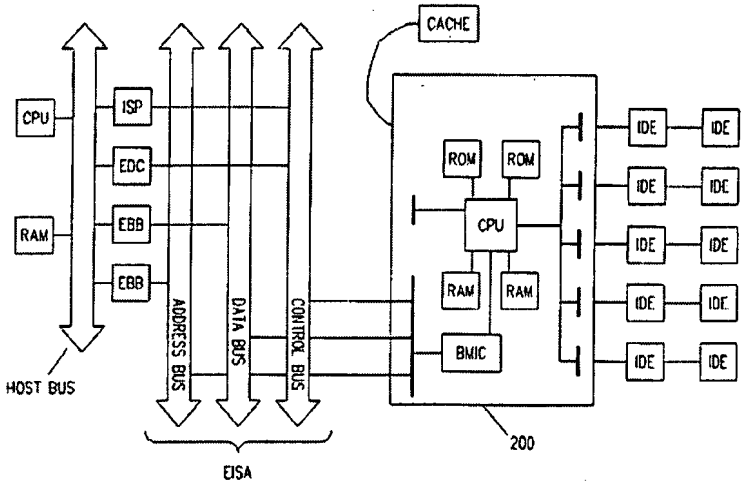
Claim Language	IEEE Standard 1212.1
	 Figure 6.1—Dispatch List structures  Figure 8.1—Mailbox DMA structures
12. The data processing system according to claim 10, wherein the non-	Higher priority processor requests are processed before lower priority requests. The requests are non-blocking. *See generally,*

Invalidity Claim Chart for U.S. Patent No. 5,812,799 - IEEE Standard 1212.1

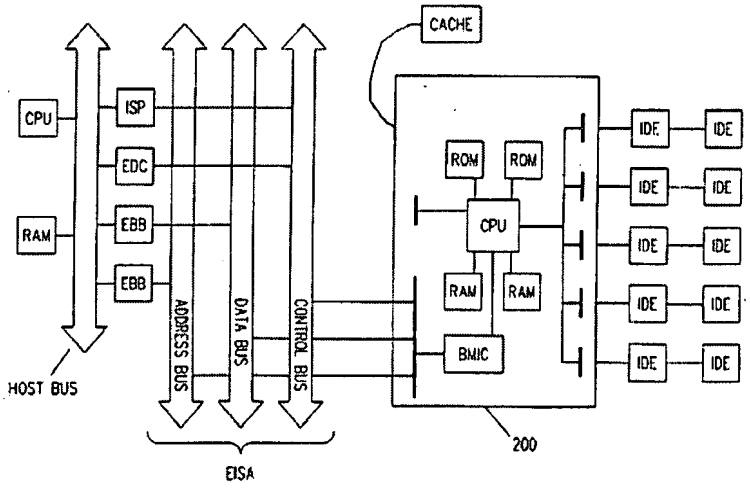
Claim Language	IEEE Standard 1212.1
blocking processor requests are prioritized in the pending queues such that the higher priority non-blocking processor requests are processed before the lower priority non-blocking processor requests.	*IEEE Standard 1212.1* at 2, 8, 11.

EXHIBIT H-2

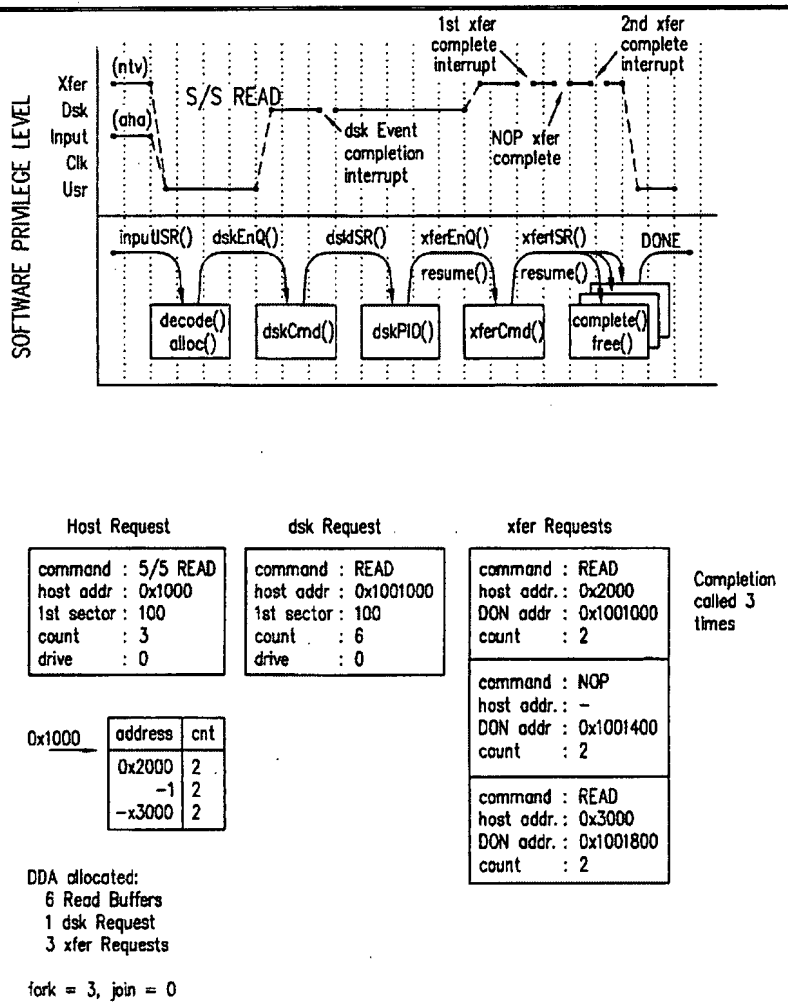
Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Parks

Claim Language	Parks
1. A data processing system comprising:	*Parks* discloses a data processing system. *See generally, Parks*, 14:13-40, 15:8-22, 20:66 to 21:3, Figs. 1, 4.  FIG. 1  FIG. 4
one or more requesting processors that generate processor requests directed to one or more peripheral devices;	*Parks* discloses a host computer with a processor that generates disk requests (i.e., processor requests) to multiple IDE disk drives and/or multiple arrays of IDE disk drives (i.e., peripheral devices) as shown in Fig. 4. *See generally, Parks*, 14:13-40, 58:9-19, Figs. 1, 4, 13.

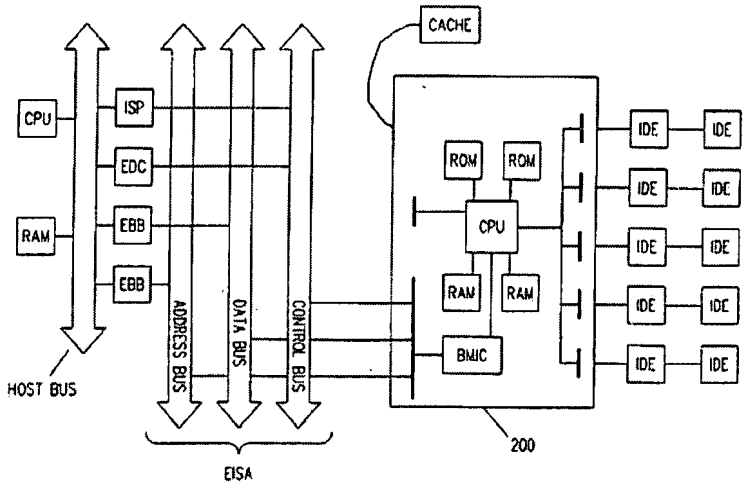
Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Parks

Claim Language	Parks
	 FIG. 4
a plurality of peripheral devices that accept the processor requests; and	*Parks* discloses multiple IDE disk drives and arrays of IDE disk drives that receive requests from the host computer. A request from the host to read from a disk is shown in Figure 13. *See generally, Parks*, 14:13-18, 22:23-30, 58:9-12, Figs. 1, 4.

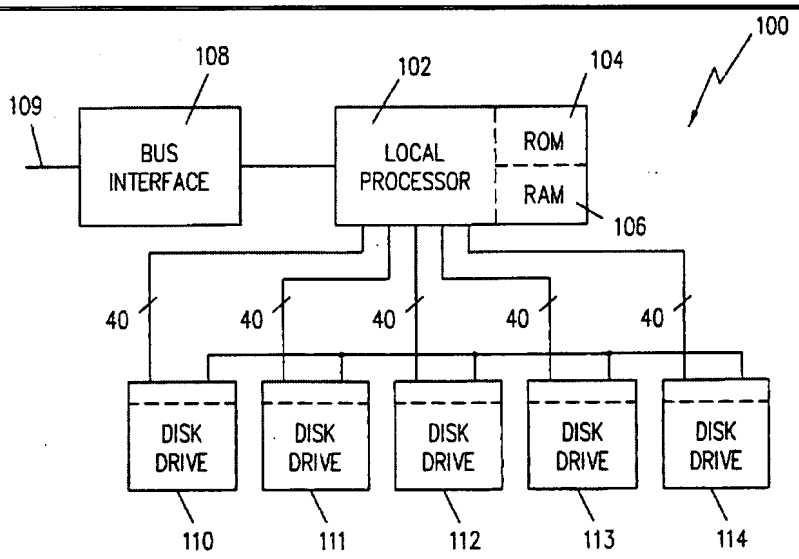
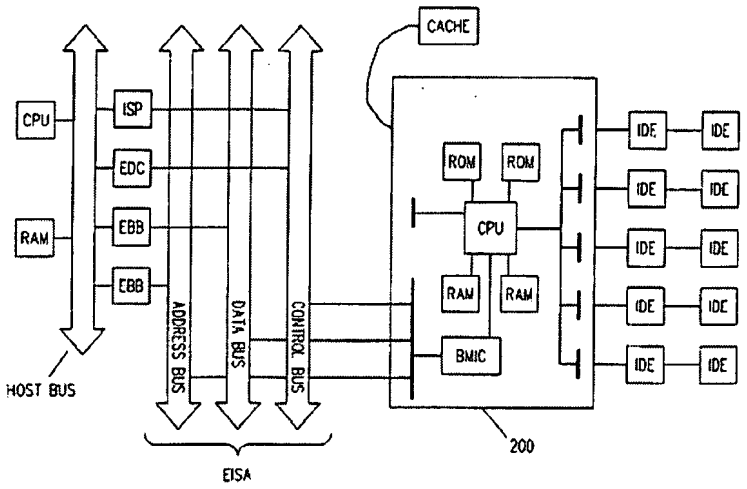
Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Parks

Claim Language	Parks
	 FIG. 13
a controller responsive to the processor requests that creates a plurality of separate pending queues corresponding to each one of the plurality of peripheral devices for queuing the processor requests directed to a particular peripheral device in entries of a corresponding separate	Controller 200 supports multiple controllers, each containing at least one queue to hold disk requests directed to corresponding IDE disk drives. *See generally, Parks*, 21:3-4, 22:23-56, 25:17-19, 25:62-66, 26:16-19.

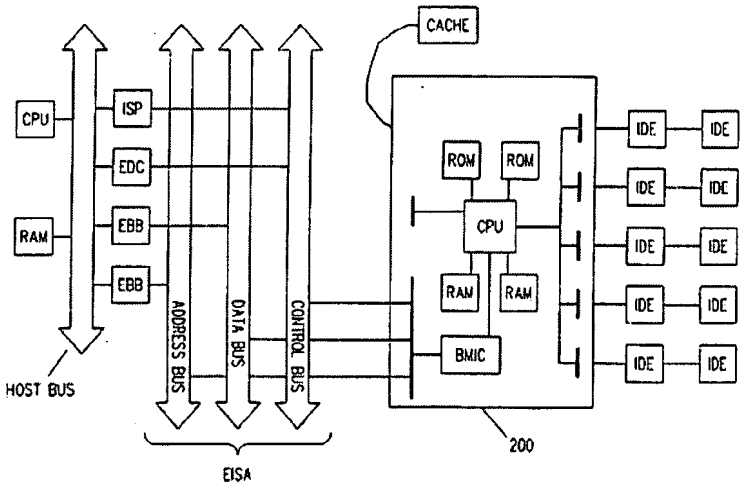
Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Parks

Claim Language	Parks
pending queue,	 FIG. 4
wherein at least two separate peripheral devices process the processor requests simultaneously, after retrieving such processor requests from their respective separate pending queues,	Controller 200 supports multiple outstanding I/O requests on each drive, with operations on separate drives occurring simultaneously. *See generally, Parks*, 17:65 to 19:4, 22:23-30, 26:16-19, 36:27-31.
wherein the one or more requesting processors include dependency checking logic that generate non-blocking processor requests.	Controller 200 supports multiple outstanding I/Os on each drive with operations on separate drives occurring simultaneously. Controller 200 promotes read requests past write requests unless the read request is for a block that the write is going to write to (i.e., checking dependencies). *See generally, Parks*, 36:29-31, 37:9-12.
3. The data processing system according to claim 1, wherein the non-blocking processor requests are generated by running real-time processes.	The data processing system can handle requests regardless of their origin, and therefore the requests can inherently be generated by running real-time processes. *See generally, Parks*, Fig. 4.
10. A data processing system comprising:	*Parks* discloses a data processing system. *See generally, Parks*, 14:13-40, 15:8-22, 20:66 to 21:3, Figs. 1, 4.

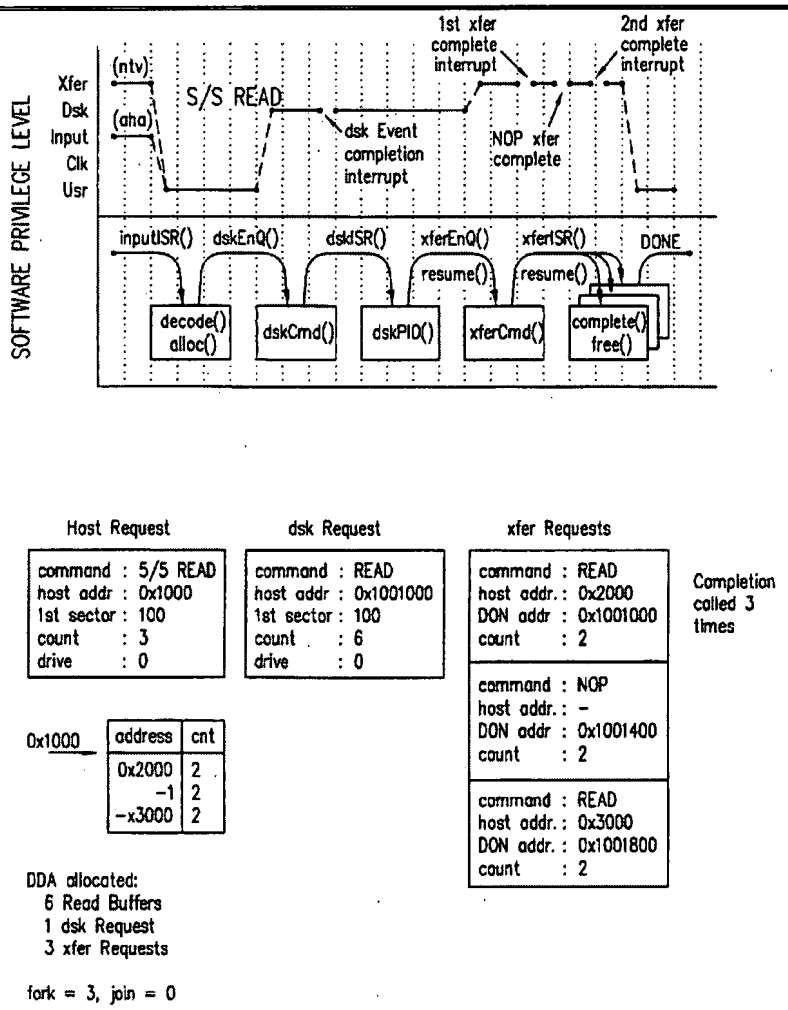
Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Parks

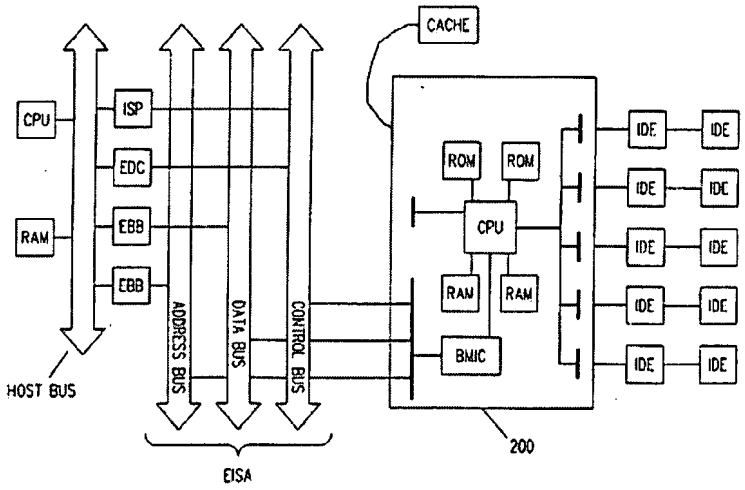
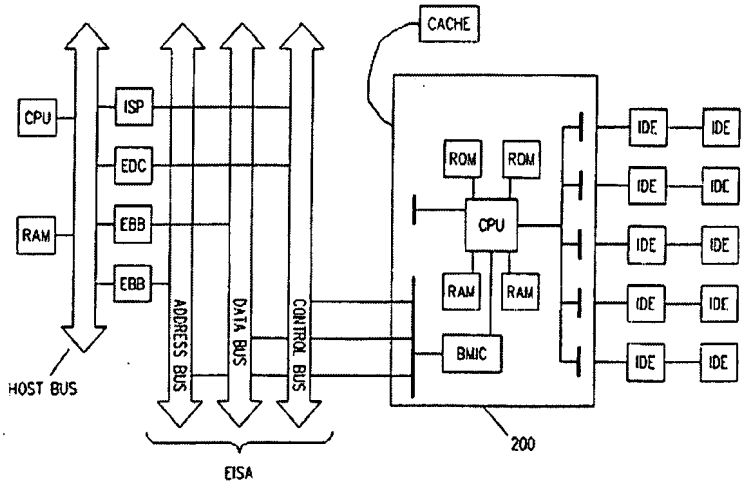
Claim Language	Parks
	 FIG. 1  FIG. 4
one or more requesting processors that generate non-blocking processor requests directed to one or more peripheral devices;	*Parks* discloses a host computer with a processor that generates disk requests (i.e., processor requests) to multiple IDE disk drives and/or multiple arrays of IDE disk drives (i.e., peripheral devices) as shown in Fig. 4. *See generally, Parks*, 14:13-40, 58:9-19, Figs. 1, 4, 13.

Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Parks

Claim Language	Parks
	 FIG. 4
a plurality of peripheral devices that accept the non-blocking processor requests;	*Parks* discloses multiple IDE disk drives and arrays of IDE disk drives that receive requests from the host computer. A request from the host to read from a disk is shown in Figure 13. *See generally, Parks*, 14:13-18, 22:23-30, 58:9-12, Figs. 1, 4.

Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Parks

Claim Language	Parks
	 FIG. 13
a shared memory device; and	RAM is a shared memory device in Controller 200 for disk drives to read and write data from (shown in Figure 4). *See generally, Parks*, Fig. 4.

Claim Language	Parks
	 FIG. 4
a memory controller responsive to the non-blocking processor requests that creates a plurality of separate pending queues on the shared memory device corresponding to each one of the plurality of peripheral devices for queuing the non-blocking processor requests directed to a particular peripheral device in entries of a corresponding separate pending queue,	As shown in Figure 4, *Parks* describes a controller 200 that supports multiple controllers, each containing at least one queue to hold disk requests directed to corresponding IDE disk drives. *See generally, Parks*, 21:3-4, 22:23-56, 25:17-19, 25:62-66, 26:16-19.  FIG. 4
wherein at least two separate peripheral devices process the non-blocking processor requests simultaneously, after retrieving such non-blocking processor	Controller 200 supports multiple outstanding I/O requests on each drive, with operations on separate drives occurring simultaneously. *See generally, Parks*, 17:65 to 19:4, 22:23-30, 26:16-19, 36:27-31.

Claim Language	Parks
requests from their respective separate pending queues,	
wherein the entries include pointers that point to memory locations on the shared memory device.	Controller 200 gets a data buffer using "GetWriteBuffer()" from the shared memory RAM and assigns a variable in an entry for a queued I/O request to an IDE disk drive, to point to the buffer. *See generally, Parks*, col. 29, 26:12-16, Fig. 4.
12. The data processing system according to claim 10, wherein the non-blocking processor requests are prioritized in the pending queues such that the higher priority non-blocking processor requests are processed before the lower priority non-blocking processor requests.	Read requests are promoted past write requests unless the read is for a block that the write is going to write to. *See generally, Parks*, 37:9-11.

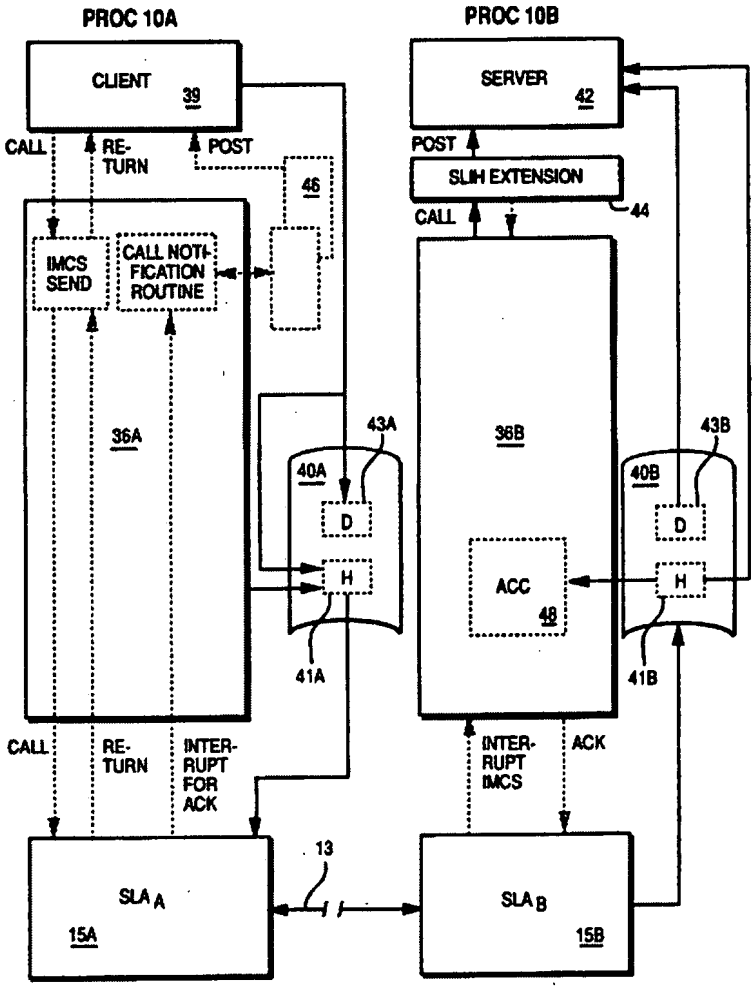
EXHIBIT H-3

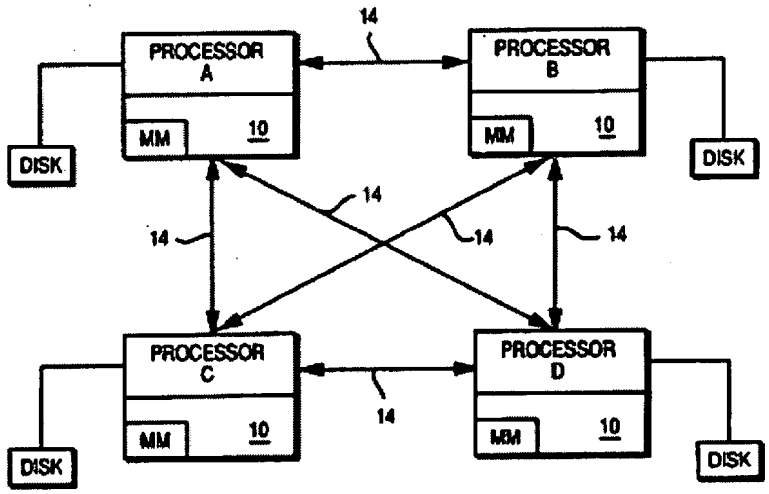
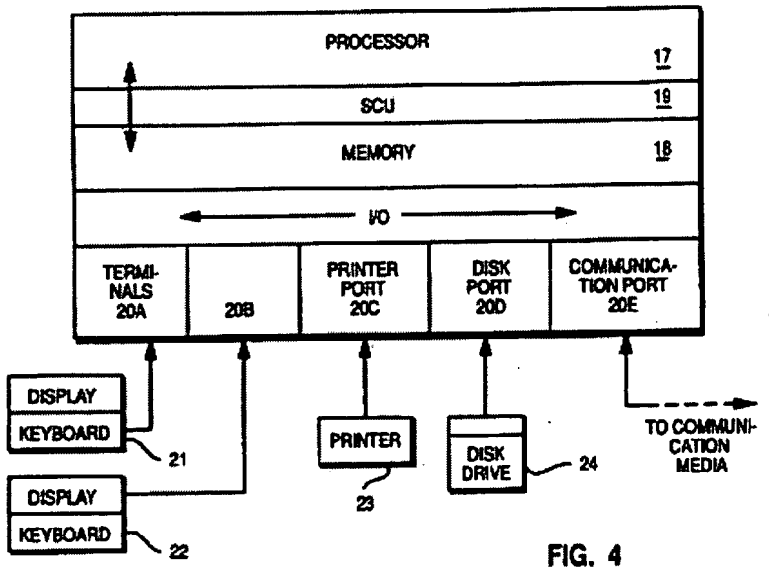
Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Sprunt

Claim Language	Sprunt
3. The data processing system according to claim 1, wherein the non-blocking processor requests are generated by running real-time processes.	*Sprunt* discloses non-blocking processor requests generated by real-time processes, such as periodic task sets for real-time systems. *See generally, Sprunt* at 152.

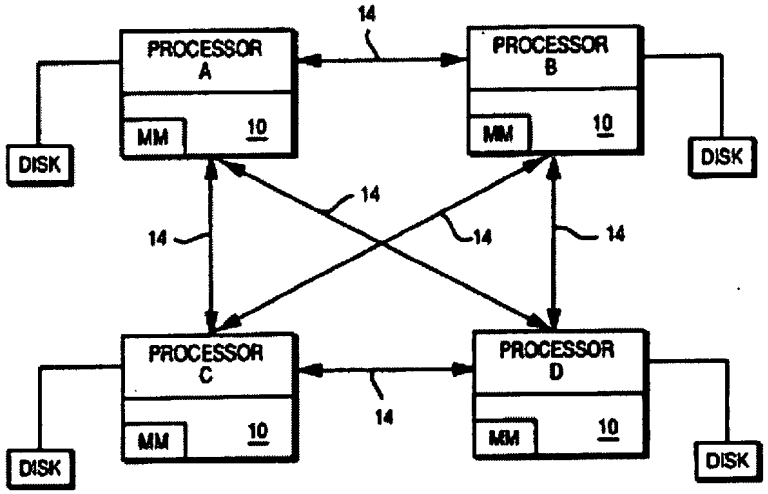
EXHIBIT H-4

Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Blount

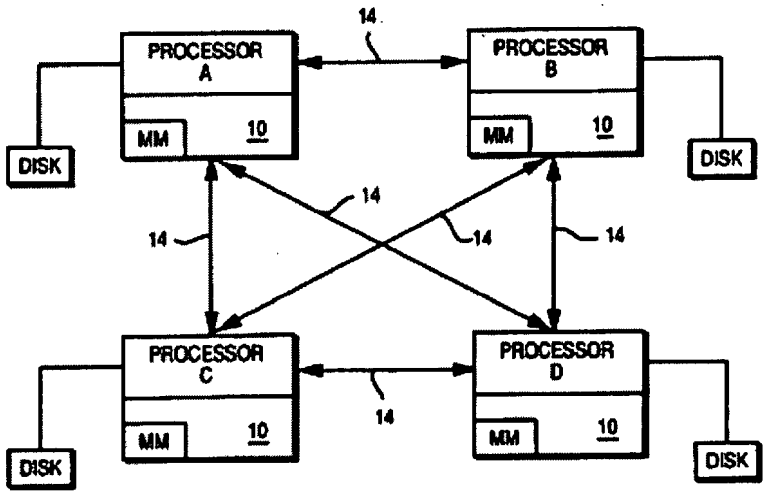
Claim Language	Blount
1. A data processing system comprising:	*Blount* discloses communication protocols for a data processing system shown in Figure 7A. *See generally, Blount*, 1:15-17, Fig. 7A.  FIG. 7A
one or more requesting processors that generate processor requests directed to one or more peripheral devices;	*Blount* discloses processor units 10A-10D, shown in one embodiment of the invention in Figure 2, that generate requests to peripheral devices such as the memories of the other processor units (shown in Figure 4). *See generally, Blount*, 3:7-15, 7:45-50, Figs. 2, 4.

Claim Language	Blount
	 FIG. 2  FIG. 4
a plurality of peripheral devices that accept the processor requests; and	*Blount* discloses multiple memories (i.e., peripheral devices) in processing units, which accept processor requests. *See generally, Blount*, 5:10-20, 7:45-50, 10:36 to 11:7, Fig. 2.

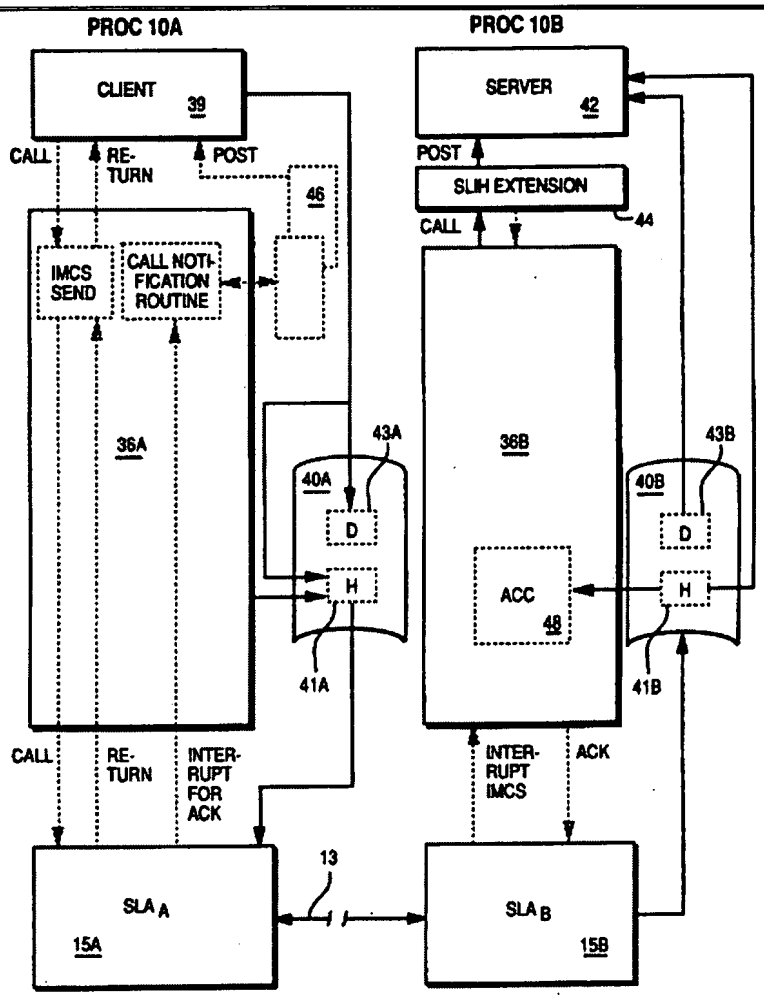
Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Blount

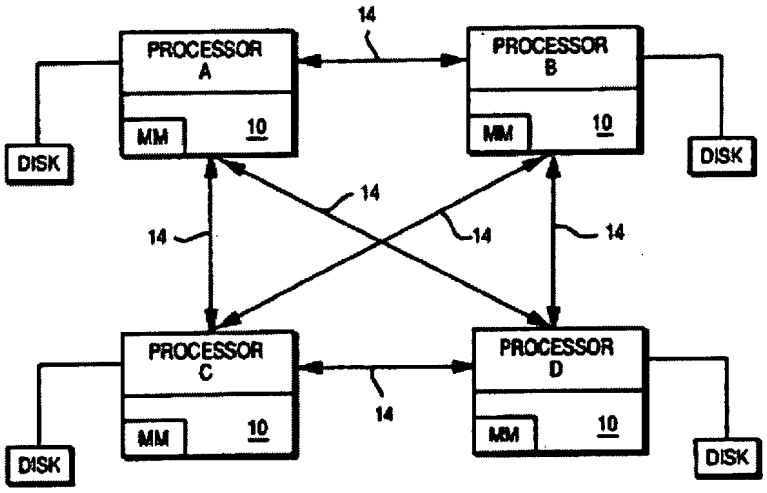
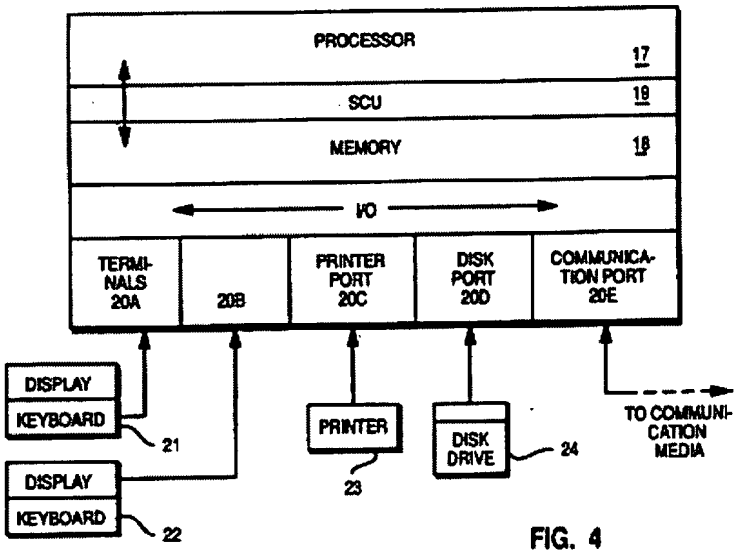
Claim Language	Blount
	 FIG. 2
a controller responsive to the processor requests that creates a plurality of separate pending queues corresponding to each one of the plurality of peripheral devices for queuing the processor requests directed to a particular peripheral device in entries of a corresponding separate pending queue,	IMCS 36 maintains separate queues in each processor unit 10A-10D to queue memory requests directed to memory (MM) at each of processor units 10A-10D. *See generally, Blount*, 9:27-41, 10:30-35, 15:52-61.
wherein at least two separate peripheral devices process the processor requests simultaneously, after retrieving such processor requests from their respective separate pending queues,	*Blount* discloses managing concurrent requests in which one unit wants to write to a particular page at the same time another unit wants to read from it. *See generally, Blount*, 4:45 to 5:2, 15:66-68, 7:11-15, 7:23-28. The requests are received on communication links like communication link 14 shown in Figure 2 and the message may be queued in a page-in request queue from which it is later taken by the server for processing or for copying the relevant data into its private data area (i.e., retrieving processor request). *See generally, Blount*, 7:23-28, 10:30-35, 10:51-56, 15:55-58, 16:4-19, 17:3-11, Fig. 2.

Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Blount

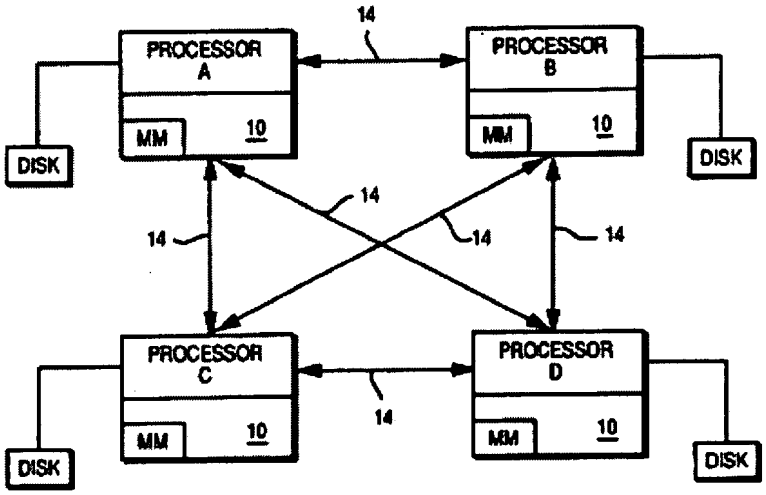
Claim Language	Blount
	 FIG. 2
wherein the one or more requesting processors include dependency checking logic that generate non-blocking processor requests.	*Blount* discloses a general purpose processor, which can inherently run software for dependency checking. *See generally, Blount*, 1:68 to 6:27.
3. The data processing system according to claim 1, wherein the non-blocking processor requests are generated by running real-time processes.	The data processing system can handle requests regardless of their origin, and therefore the requests can inherently be generated by running real-time processes. *See generally, Blount*, Figs. 2, 7A, 7B.
10. A data processing system comprising:	*Blount* discloses communication protocols for a data processing system shown in Figure 7A. *See generally, Blount*, 1:15-17, Fig. 7A.

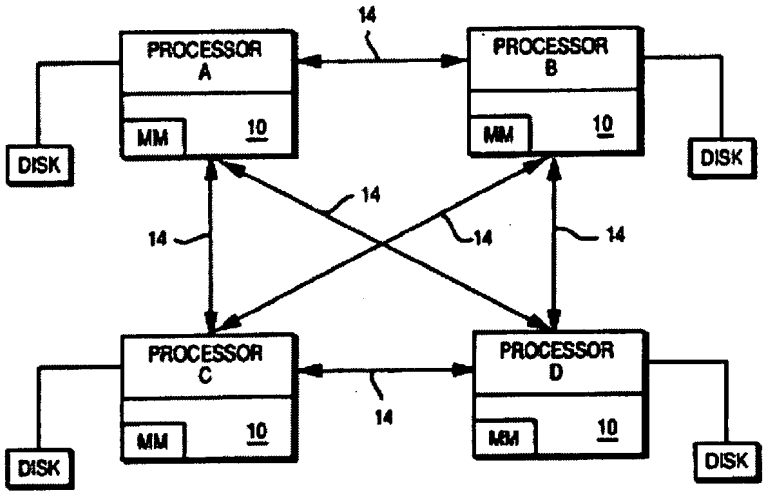
Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Blount

Claim Language	Blount
	 FIG. 7A
one or more requesting processors that generate non-blocking processor requests directed to one or more peripheral devices;	*Blount* discloses processor units 10A-10D, shown in one embodiment of the invention in Figure 2, that generate requests to peripheral devices such as the memories of the other processor units, as shown in Figure 4. *See generally, Blount*, 3:7-15, 7:45-50, Figs. 2, 4.

Claim Language	Blount
	 FIG. 2  FIG. 4 The server will service concurrent requests. *See generally, Blount*, 4:45 to 5:2, 15:66-68, 18:34-46.
a plurality of peripheral devices that accept the non-blocking processor requests;	*Blount* discloses multiple memories from processing units (i.e., peripheral devices) which accept processor requests. Figure 2, an embodiment of the invention, shows memory (MM) located within multiple processing units. *See generally, Blount*, 5:10-20, 7:45-50, 10:36 to 11:7, Fig. 2.

Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Blount

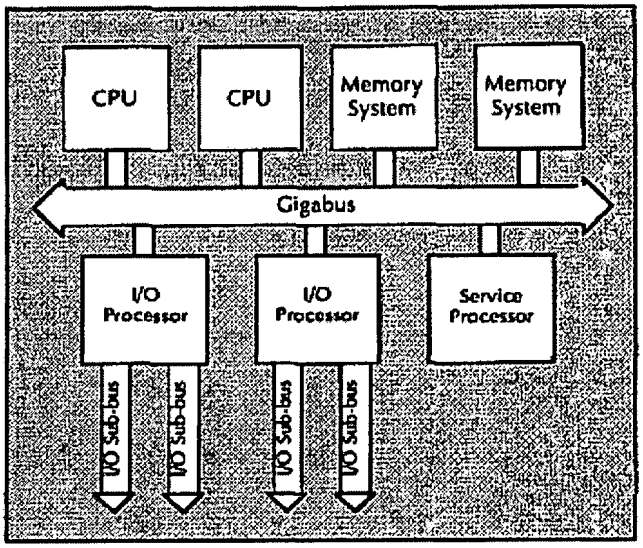
Claim Language	Blount
	 FIG. 2 The server will service concurrent requests. *See generally, Blount*, 4:45 to 5:2, 15:66-68, 18:34-46.
a shared memory device; and	*Blount* discloses a data buffer pool 43 that is shared between the servers, and is located on a shared memory device. *See generally, Blount*, 3:7-15, 11:8-12, 14:50-60, Fig. 7A.
a memory controller responsive to the non-blocking processor requests that creates a plurality of separate pending queues on the shared memory device corresponding to each one of the plurality of peripheral devices for queuing the non-blocking processor requests directed to a particular peripheral device in entries of a corresponding separate pending queue,	IMCS 36 maintains separate queues in each processor unit 10A-10D to queue memory requests directed to memory (MM) at each of processor units 10A-10D. *See generally, Blount*, 9:27-41, 10:30-35, 15:52-61. The server will service concurrent requests. *See generally, Blount*, 4:45 to 5:2, 15:66-68,18:34-46.
wherein at least two separate peripheral devices process the non-blocking processor requests	*Blount* discloses managing concurrent requests in which one unit wants to write to a particular page at the same time another unit wants to read from it. *See generally, Blount*, 4:52 to 5:2, 7:11-15, 7:23-28, 15:66-68.

Claim Language	Blount
simultaneously, after retrieving such non-blocking processor requests from their respective separate pending queues,	The requests are received on communication links like communication link 14 shown in Figure 2 and the message may be queued in a page-in request queue from which it is later taken by the server for processing or for copying the relevant data into its private data area (i.e., retrieving processor request). *See generally, Blount*, 7:23-28, 10:30-35, 10:51-56, 15:55-58, 16:4-19, 17:3-11, Fig. 2.  FIG. 2 The server will service concurrent requests. *See generally, Blount*, 4:45 to 5:2, 15:66-68, 18:34-46.
wherein the entries include pointers that point to memory locations on the shared memory device.	Entries contain data and control information and the control information can include pointers to the data to be read or written. *See generally, Blount*, 10:21-29, 10:39-42. *Blount* teaches that a data buffer pool 43 is shared between the servers. *See generally, Blount*, 3:7-15, 14:50-60, Fig. 7A.
12. The data processing system according to claim 10, wherein the non-blocking processor requests are prioritized in the pending queues such that the higher priority non-blocking processor requests are processed before the lower priority non-blocking processor	*Blount* teaches that higher priority requests are processed before lower priority requests. *See generally, Blount*, 4:45 to 5:2, 14:37-46, 15:52-61.

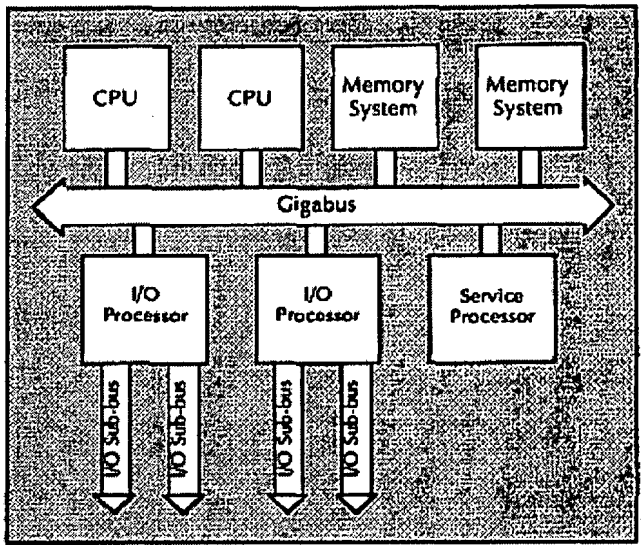
Claim Language	Blount
requests.	

EXHIBIT H-5

Invalidity Claim Chart for U.S. Patent No. 5,812,799 - ELXSI

Claim Language	ELXSI
1. A data processing system comprising:	The ELXSI System is a data processing system. *See generally, ELXSI System Foundation Guide*, Fig. 2-1; *ELXSI System Architecture* at 1-1 to 1-4.  **Figure 2-1. Components of the ELXSI System 6400**
one or more requesting processors that generate processor requests directed to one or more peripheral devices;	The ELXSI System has multiple requesting processors that generate requests (messages) to multiple peripheral devices. *See generally, ELXSI System Foundation Guide* at 2-1, 2-7, 2-19 to 2-20, 6-18, Fig. 2-1; *ELXSI System Architecture* at 1-1 to 1-4.
a plurality of peripheral devices that accept the processor requests; and	The peripheral devices accept the processor requests (messages). *See generally, ELXSI System Foundation Guide* at 5-5 to 5-6; *ELXSI System Architecture* at 1-1 to 1-4, 13-7 to 13-8.
a controller responsive to the processor requests that creates a plurality of separate pending queues corresponding to each one of the plurality of peripheral devices for queuing the processor requests directed to a particular peripheral device	The ELXSI System has a controller that creates separate pending queues (funnels) that correspond to each one of the plurality of peripheral devices for queuing processor requests (messages). *See generally, ELXSI System Foundation Guide* at 2-2, 2-5, 2-17 to 2-20, 5-1 to 5-2, 5-5, 5-5 n. 3; *ELXSI System Architecture* at 13-2 to 13-4.

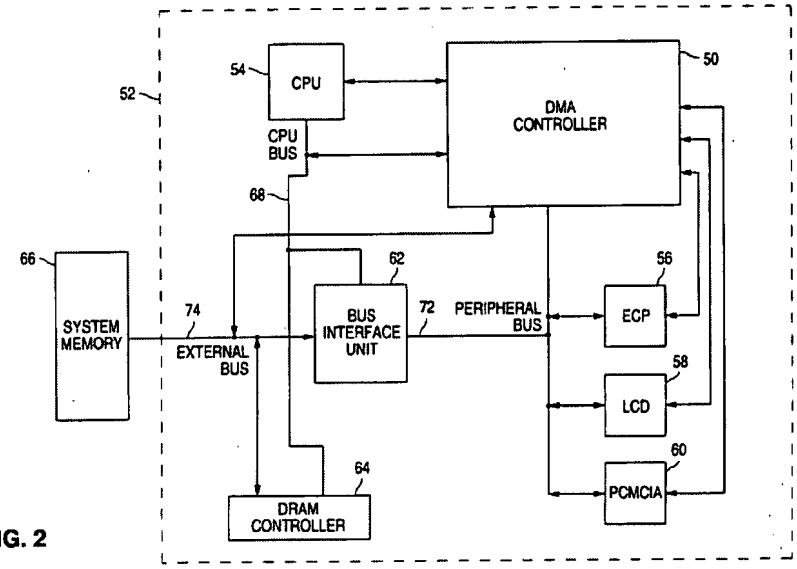
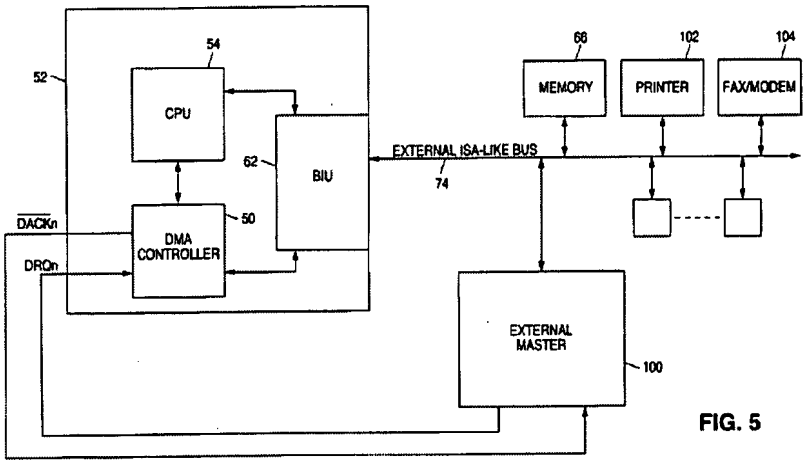
Claim Language	ELXSI
in entries of a corresponding separate pending queue,	
wherein at least two separate peripheral devices process the processor requests simultaneously, after retrieving such processor requests from their respective separate pending queues,	The peripheral devices simultaneously process requests (messages) after retrieving them from their respective pending queues (funnels). *See generally, ELXSI System Foundation Guide* at 5-5 to 5-6, 6-1; *ELXSI System Architecture* at 1-1, 13-7 to 13-8.
wherein the one or more requesting processors include dependency checking logic that generate non-blocking processor requests.	The requesting processors have dependency checking logic that generates non-blocking processor requests. *See generally, ELXSI System Foundation Guide* at 4-18 to 4-19, *ELXSI System Architecture* at 1-1.
3. The data processing system according to claim 1, wherein the non-blocking processor requests are generated by running real-time processes.	The requesting processors generate non-blocking requests by running real-time processes. *See generally, ELXSI System Foundation Guide* at 1-1, 1-4 to 1-6, 3-5; *ELXSI System Architecture* at 1-1.
10. A data processing system comprising:	The ELXSI System is a data processing system. *See generally, ELXSI System Foundation Guide*, Fig. 2-1; *ELXSI System Architecture* at 1-1 to 1-4.

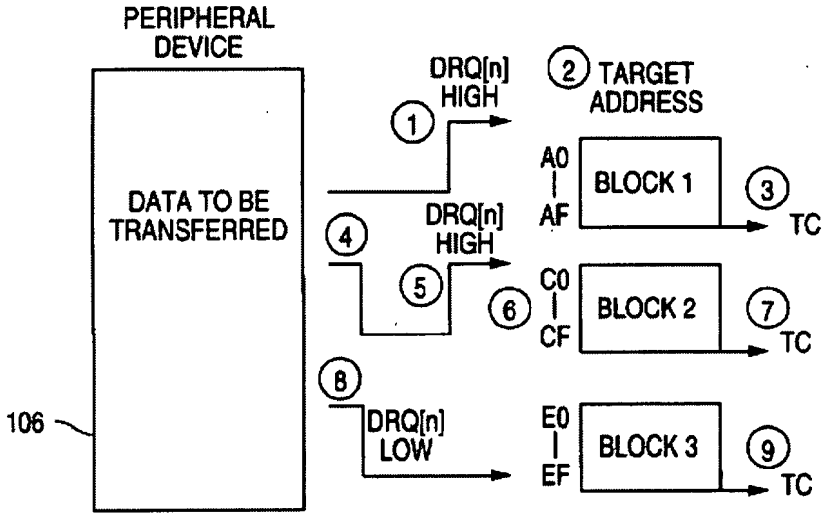
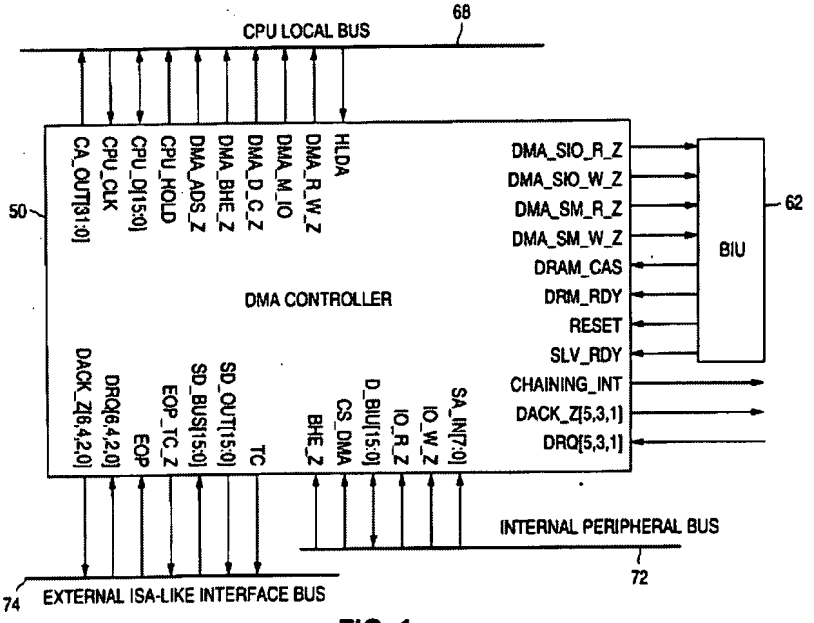
Claim Language	ELXSI
	 **Figure 2-1. Components of the ELXSI System 6400**
one or more requesting processors that generate non-blocking processor requests directed to one or more peripheral devices;	The ELXSI System has multiple requesting processors that generate non-blocking requests (messages) to multiple peripheral devices. *See generally, ELXSI System Foundation Guide* at 2-1, 2-7, 2-19 to 2-20, 6-18, Fig. 2-1; *ELXSI System Architecture* at 1-1 to 1-4.
a plurality of peripheral devices that accept the non-blocking processor requests;	The peripheral devices accept the processor requests (messages). *See generally, ELXSI System Foundation Guide* at 5-5 to 5-6; *ELXSI System Architecture* at 1-1 to 1-4, 13-7 to 13-8.
a shared memory device; and	The ELXSI System has a shared memory device. *See generally, ELXSI System Foundation Guide* at 5-5, 5-5 n.3, 5-22.
a memory controller responsive to the non-blocking processor requests that creates a plurality of separate pending queues on the shared memory device corresponding to each one of the plurality of peripheral devices for queuing the non-blocking	The ELXSI System has a controller that creates separate pending queues (funnels) that correspond to each one of the plurality of peripheral devices for queuing processor requests (messages). *See generally, ELXSI System Foundation Guide* at 2-2, 2-5, 2-17 to 2-20, 5-1 to 5-2, 5-5, 5-5 n. 3; *ELXSI System Architecture* at 13-2 to 13-4.

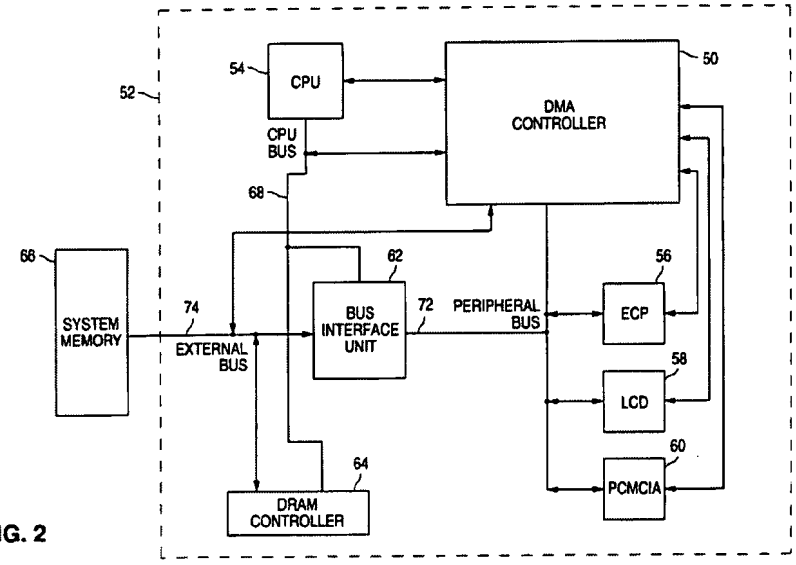
Claim Language	ELXSI
processor requests directed to a particular peripheral device in entries of a corresponding separate pending queue,	
wherein at least two separate peripheral devices process the non-blocking processor requests simultaneously, after retrieving such non-blocking processor requests from their respective separate pending queues,	The peripheral devices simultaneously process requests (messages) after retrieving them from their respective pending queues (funnels). *See generally, ELXSI System Foundation Guide* at 5-5 to 5-6, 6-1; *ELXSI System Architecture* at 1-1, 13-7 to 13-8.
wherein the entries include pointers that point to memory locations on the shared memory device.	
12. The data processing system according to claim 10, wherein the non-blocking processor requests are prioritized in the pending queues such that the higher priority non-blocking processor requests are processed before the lower priority non-blocking processor requests.	The higher priority processor requests (messages) are processed before the lower priority processor requests. *See generally, ELXSI System Foundation Guide* at 3-17, 3-17 n. 1, 5-11; *ELXSI System Architecture* at 13-26 to 13-28.

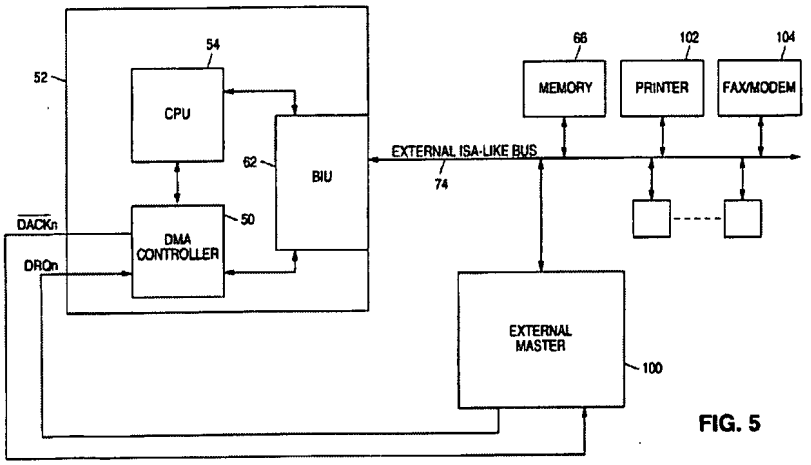
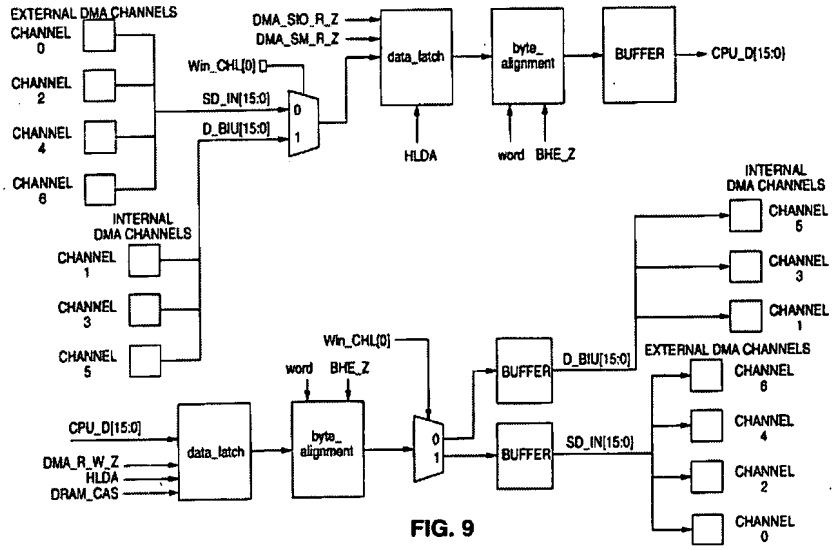
EXHIBIT H-6

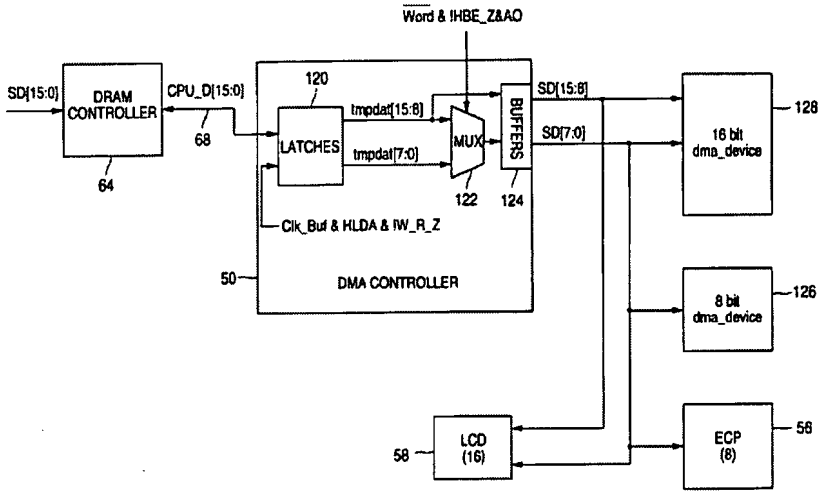
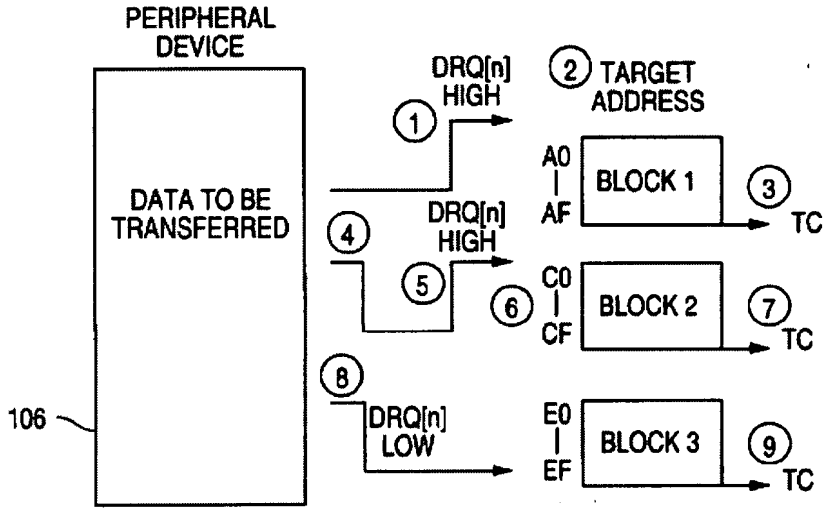
Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Pang

Claim Language	Pang
1. A data processing system comprising:	*Pang* discloses a data processing system. *See generally, Pang*, Fig. 2.  FIG. 2
one or more requesting processors that generate processor requests directed to one or more peripheral devices;	A requesting processor (CPU) generates requests to peripheral devices. *See generally, Pang*, 1:21-23, 4:7-11, 4:53-65, Figs. 2, 5.  FIG. 5
a plurality of peripheral devices that accept the processor requests; and	The peripheral devices accept processor requests. *See generally, Pang*, 4:7-11, 6:22-24, 13:25-31, Figs. 2, 5.
a controller responsive to the processor requests that creates a plurality of	A controller responsive to processor requests creates a plurality of separate pending queues (buffer chaining mode lists) that correspond to the peripheral devices. *See generally, Pang*, 1:62-

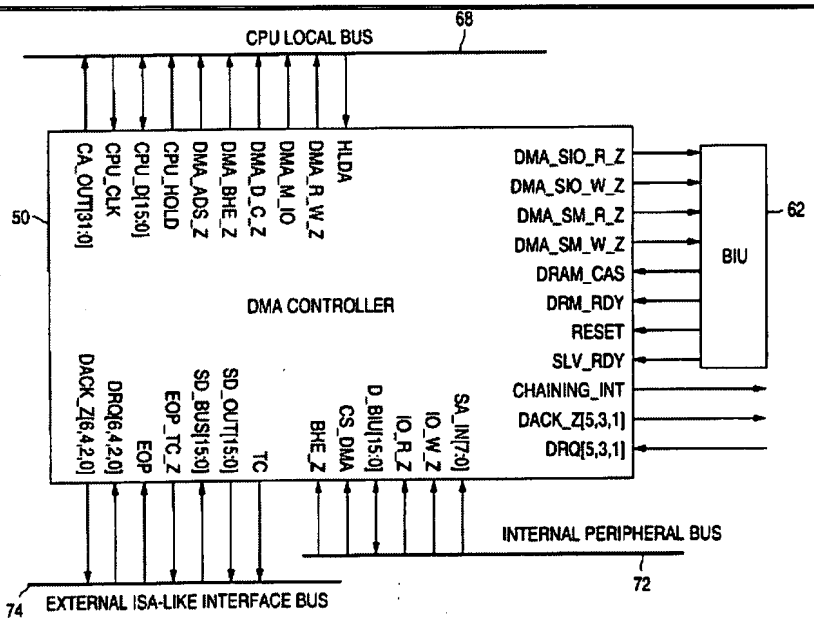
Claim Language	Pang
separate pending queues corresponding to each one of the plurality of peripheral devices for queuing the processor requests directed to a particular peripheral device in entries of a corresponding separate pending queue,	67, 4:53-65, 5:14-22, 13:46-60, Figs. 1, 2, 11.  **FIG. 11**  **FIG. 1**
wherein at least two separate peripheral devices process the processor requests simultaneously, after retrieving such processor	Peripheral devices retrieve and then process requests from their respective pending queues (buffer chaining mode lists). *See generally, Pang*, 6:22-24, 13:25-31.

Claim Language	Pang
requests from their respective separate pending queues,	
wherein the one or more requesting processors include dependency checking logic that generate non-blocking processor requests.	The processor (CPU) is a general purpose processor, which inherently has software for dependency checking. *See generally, Pang*, Figs. 2, 5.
3. The data processing system according to claim 1, wherein the non-blocking processor requests are generated by running real-time processes.	The data processing system can handle requests regardless of their origin, and therefore the requests can inherently be generated by running real-time processes. *See generally, Pang*, Figs. 2, 5.
10. A data processing system comprising:	*Pang* discloses a data processing system. *See generally, Pang*, Fig. 2.  FIG. 2
one or more requesting processors that generate non-blocking processor requests directed to one	A requesting processor (CPU) generates requests to peripheral devices. *See generally, Pang*, 1:21-23, 4:7-11, 4:53-65, Figs. 2, 5. The processor requests are non-blocking. *See generally, Pang*,

Claim Language	Pang
or more peripheral devices;	1:18-21.  FIG. 5
a plurality of peripheral devices that accept the non-blocking processor requests;	The peripheral devices accept processor requests. *See generally, Pang*, 4:7-11, 6:22-24, 13:25-31, Figs. 2, 5.
a shared memory device; and	A controller generates the pending queues (buffer chaining mode lists) from a free pool of buffers on shared memory. *See generally, Pang*, 12:12-50, Figs. 9, 10.  FIG. 9

Claim Language	Pang
	 FIG. 10
a memory controller responsive to the non-blocking processor requests that creates a plurality of separate pending queues on the shared memory device corresponding to each one of the plurality of peripheral devices for queuing the non-blocking processor requests directed to a particular peripheral device in entries of a corresponding separate pending queue,	A controller responsive to processor requests creates a plurality of separate pending queues (buffer chaining mode lists) that correspond to the peripheral devices. *See generally, Pang*, 1:62-67, 4:53-65, 5:14-22, 13:46-60, Figs. 1, 2, 11.  FIG. 11

Invalidity Claim Chart for U.S. Patent No. 5,812,799 - Pang

Claim Language	Pang
	 **FIG. 1**
wherein at least two separate peripheral devices process the non-blocking processor requests simultaneously, after retrieving such non-blocking processor requests from their respective separate pending queues,	Peripheral devices retrieve and then process requests from their respective pending queues (buffer chaining mode lists). *See generally, Pang*, 6:22-24, 13:25-31.
wherein the entries include pointers that point to memory locations on the shared memory device.	
12. The data processing system according to claim 10, wherein the non-blocking processor requests are prioritized in the pending queues such that the higher priority non-blocking processor	The processor requests are prioritized in the pending queues such that the higher priority requests are at least sometimes processed before the lower priority requests. *See generally, Pang*, 4:58-63, 9:60-67.

Claim Language	Pang
requests are processed before the lower priority non-blocking processor requests.	

Table 1

Abbreviated Name	Full Citation / Note
Blount	U.S. Patent No. 5,253,342 (Oct. 12, 1993).
ELXSI System 6400	Described by, for example, *ELXSI System Foundation Guide* and *ELXSI System Architecture.*
ELXSI System Architecture	"System Architecture," ELXSI (2d Ed. Oct. 1983).
ELXSI System Foundation Guide	"System Foundation Guide," ELXSI (1st Ed. Oct. 1987).
IEEE Standard 1212.1	"IEEE Standard for Communicating Among Processors and Peripherals Using Shared Memory (Direct Memory Access—DMA)," IEEE (1994).
Pang	U.S. Patent No. 5,826,106 (Oct. 20, 1998).
Parks	U.S. Patent No. 5,530,960 (June 25, 1996).
PS1 System	Sony PlayStation system described by, for example, *PSX CPU User's Manual* and *PSX CPU and Peripheral Specifications.*
PSX CPU and Peripheral Specifications	"CPU and Peripheral Specifications," v. 1.3 (Nov. 18, 1994).
PSX CPU User's Manual	"PSX CPU User's Manual," v. 1.1 (1994).
Sprunt	Sprunt et al., "Priority-Driven, Preemptive I/O Controllers for Real-Time Systems," IEEE (1988).

Table 2 - Prior Art

Reference / System
Blount
ELXSI System 6400
ELXSI System Architecture
ELXSI System Foundation Guide
IEEE Standard 1212.1
Pang
Parks
PS1 System
PSX CPU and Peripheral Specifications
PSX CPU User's Manual

Table 3 - Prior Art

Reference / System
Sprunt